Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning

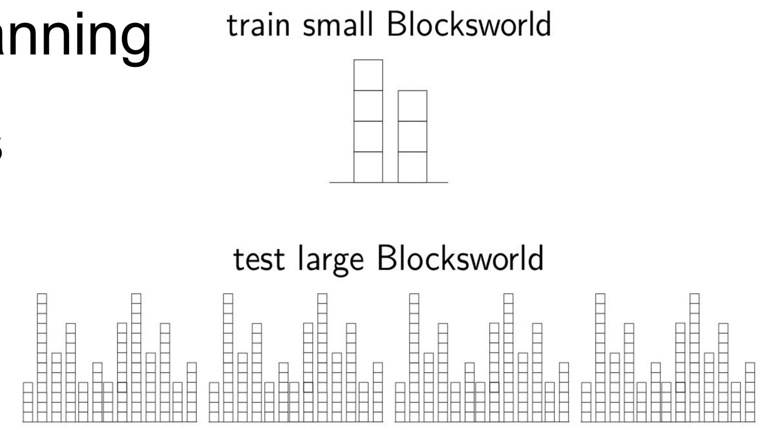
Dillon Z. Chen[1,2] Felipe Trevizan[2] Sylvie Thiébaux[1,2]

[1]LAAS-CNRS, Université de Toulouse, [2]Australian National University

dillon.chen@laas.fr, felipe.trevizan@anu.edu.au, sylvie.thiebaux@laas.fr

The problem we are solving.

- Learning for domain-specific planning
- Train models on small problems
- Test on large problems. e.g.

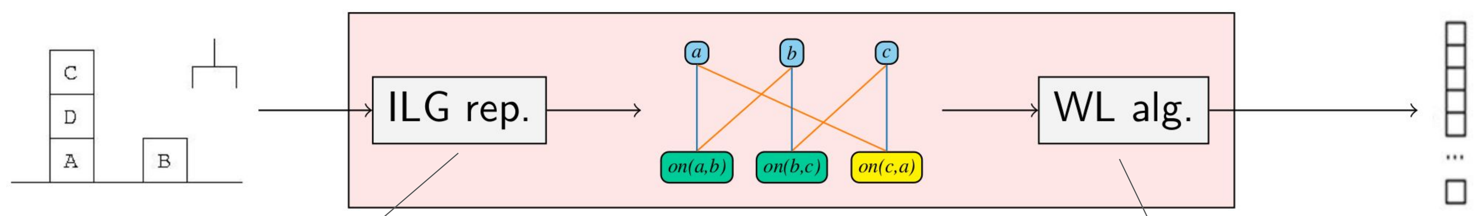


Contributions

1. Automatic feature generation for planning tasks
2. Theoretical comparisons to related work
3. Experiments

1. New WL Features for Planning

- **new:** construct *Instance Learning Graph* for states
- run Weisfeiler-Leman (WL) algorithm for edge-labelled graphs



Definition 3.1. The *instance learning graph (ILG)* of a lifted planning problem $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{A}, s_0, G \rangle$ is the graph $G = \langle V, E, c, l \rangle$ with

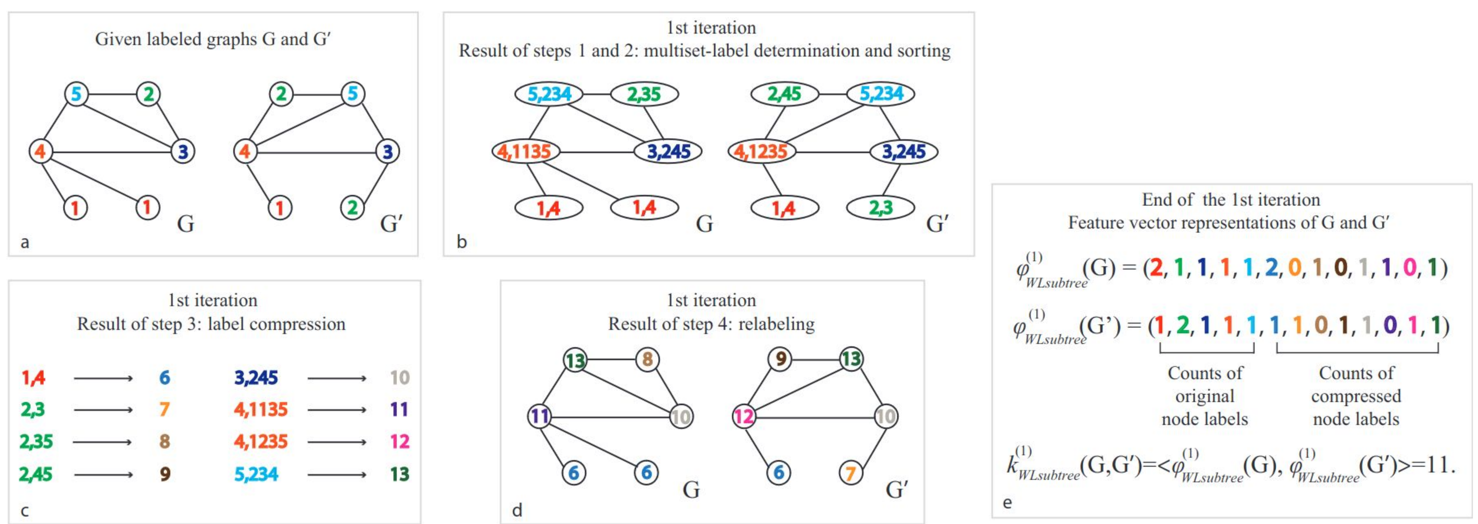
- $V = \mathcal{O} \cup s_0 \cup G$
- $E = \bigcup_{p=P(o_1,\ldots,o_{n_P}) \in s_0 \cup G} \{\langle p, o_1 \rangle, \ldots, \langle p, o_{n_P} \rangle\}$
- $c : V \to (\{\mathtt{ap}, \mathtt{ug}, \mathtt{ag}\} \times \mathcal{P}) \cup \{\mathtt{ob}\}$ defined by

$$u \mapsto \begin{cases} \mathtt{ob}, & \text{if } u \in \mathcal{O}; \\ (\mathtt{ag}, P), & \text{if } u = P(o_1, \ldots, o_{n_P}) \in s_0 \cap G; \\ (\mathtt{ap}, P), & \text{if } u = P(o_1, \ldots, o_{n_P}) \in s_0 \setminus G; \\ (\mathtt{ug}, P), & \text{if } u = P(o_1, \ldots, o_{n_P}) \in G \setminus s_0; \end{cases}$$

- $l : E \to \mathbb{N}$ with $\langle p, o_i \rangle \mapsto i$.

$c^0(v) \leftarrow c(v), \forall v \in V$
for $j = 1, \ldots, L$ **do for** $v \in V$ **do**
  $c^j(v) \leftarrow \mathrm{hash}\left(c^{j-1}(v), \bigcup_{\iota \in \Sigma_E} \{\!\{(c^{j-1}(u), \iota) \mid u \in \mathcal{N}_\iota(v)\}\!\}\right)$
return $\bigcup_{j=0,\ldots,L} \{\!\{c^j(v) \mid v \in V\}\!\}$

- WL algorithm idea: iteratively refine node colours based on neighbouring nodes (image from [3])



+ Feature vectors agnostic to downstream model
+ Fast to train (up to 900x faster than GNNs) + generate
+ Few parameters (up to 600x fewer than GNNs)
+ State-of-the-art expressivity (see **2.**)
+ Explainable features

Deep Learning for Planning is Overrated

• Neural Networks

Expensive to train and use

Non-deterministic optimisation

Not explainable

• GNNs

Research saturated on molecular datasets

Limited expressivity [1]

New models barely beat 1-WL and expensive

• Transformers

Discount GNNs + an ordering on nodes

Claims often exaggerated and overstated [2]

Poor generalisation

Explainable Features

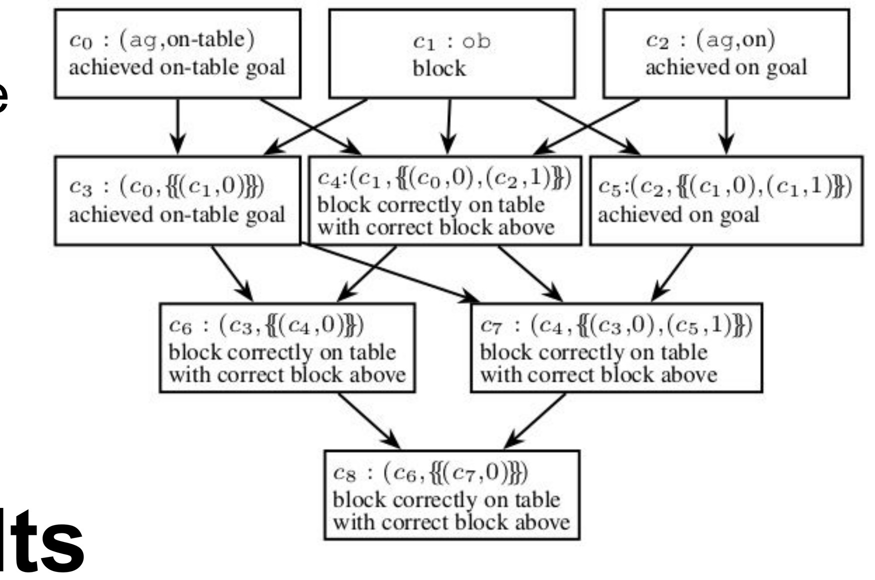
❖ features understood by analysing dependency graph

❖ e.g. Blocksworld; largest feature in trained model:

"number of blocks correctly on table with correct block above it"

❖ weight of -1.76,

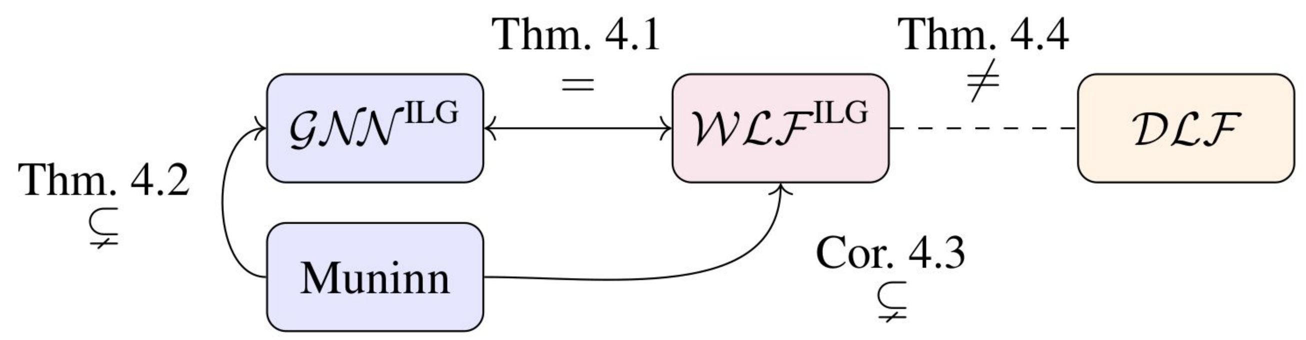
⇒ rewards correct blocks on table



2. Theoretical Results

- *WLF* = new contribution
- *DLF* = Description Logic Features [4]
- *Muninn* = Relational NNs for Planning [5]

WL Features most expressive, alongside DLF



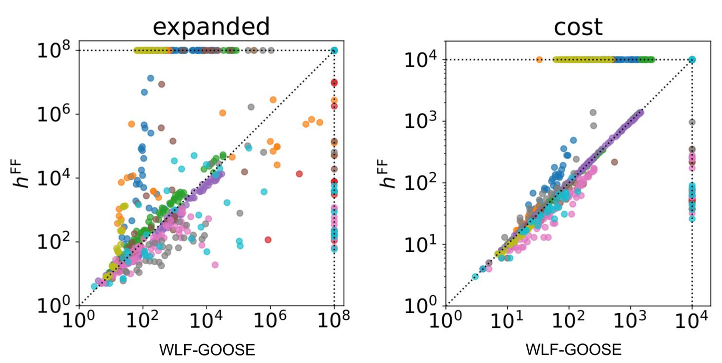
3. Experiments

- IPC 2023 Learning Track
- train and test set sizes →
- coverage results ↓ *(it works)*

domain	Training		Testing	
blocksworld	$n \in [2, 21]$	56	$n \in [5, 488]$	90
childsnack	$c \in [1, 5]$	37	$c \in [4, 292]$	90
ferry	$c \in [1, 13]$	66	$c \in [2, 974]$	90
floortile	$t \in [2, 30]$	64	$t \in [12, 980]$	90
miconic	$p \in [1, 10]$	99	$p \in [1, 485]$	90
rovers	$r \in [1, 4]$	67	$r \in [1, 30]$	90
satellite	$s \in [1, 10]$	90	$n \in [3, 99]$	90
sokoban	$n \in [7, 13]$	99	$b \in [8, 99]$	90
spanner	$s \in [1, 10]$	89	$n \in [1, 487]$	90
transport	$v \in [1, 5]$	47	$n \in [3, 50]$	90

Domain	h^{FF}	Muninn	GNN-GOOSE	WLF-GOOSE	LAMA
blocksworld	28	53	63	**75**	61
childsnack	26	12	23	**29**	35
ferry	68	38	70	**76**	68
floortile	**12**	1	0	2	11
miconic	**90**	**90**	89	**90**	90
rovers	34	24	26	**37**	67
satellite	**65**	16	31	53	89
sokoban	36	31	33	**38**	40
spanner	30	**76**	46	73	30
transport	**41**	24	32	29	66
all	430	365	413	**502**	557
IPC score	393.5	328.9	391.0	**461.3**	492.7

>20% more coverage than GNNs



[1] Ryoma Sato. *A Survey on The Expressive Power of Graph Neural Networks.* arXiv 2020

[2] Rylan Schaeffer, Brando Miranda, Sanmi Koyejo. *Are Emergent Abilities of Large Language Models a Mirage?* NeurIPS 2023

[3] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, Karsten M. Borgwardt. *Weisfeiler-Lehman Graph Kernels.* J. Mach. Learn. Res. 2011

[4] Mario Martín, Hector Geffner. *Learning Generalized Policies from Planning Examples Using Concept Languages.* Appl. Intell. 2004

[5] Simon Ståhlberg, Blai Bonet, Hector Geffner. *Learning General Optimal Policies with Graph Neural Networks: Expressive Power, Transparency, and Limits.* ICAPS 2022